CSC 261/461
Database Systems

Eustrat Zhupa

September 24, 2018

Structured Query Language

Joins

```
SELECT Fname, Lname, Address
FROM EMPLOYEE, DEPARTMENT
WHERE Dname='Research' AND Dnumber=Dno;

SELECT Fname, Lname, Address
FROM (EMPLOYEE JOIN DEPARTMENT ON Dno=Dnumber)
WHERE Dname='Research';
```

Structured Query Language

NATURAL JOIN

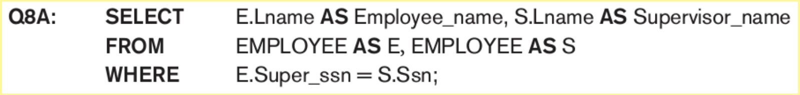
- In a *NATURAL JOIN* there is no join condition
- attributes with the same name are involved
- each such pair of attributes is included only once in the result
- If names are not the same in the base relations, then rename

```
SELECT Fname, Lname, Address
FROM (EMPLOYEE NATURAL JOIN (DEPARTMENT AS DEPT (Dname, Dno, Mssn, Msdate)
WHERE Dname='Research';
```

Structured Query Language

Joins

- The default join is an *inner join*
  - a tuple is included in the result only if a matching tuple exists in the other relation.
  - NULL values are excluded.



```
Q8A:    SELECT    E.Lname AS Employee_name, S.Lname AS Supervisor_name
        FROM      EMPLOYEE AS E, EMPLOYEE AS S
        WHERE     E.Super_ssn = S.Ssn;
```

Structured Query Language

Joins

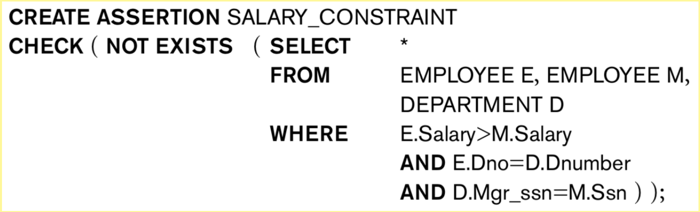
- `(INNER) JOIN`: Returns records that have matching values in both tables
- `LEFT (OUTER) JOIN`: Return all records from the left table, and the matched records from the right table
- `RIGHT (OUTER) JOIN`: Return all records from the right table, and the matched records from the left table
- `FULL (OUTER) JOIN`: Return all records when there is a match in either left or right table

SQL

- `CREATE ASSERTION`
  - used to specify additional types of constraints not covered with built-in constraints.
- `CREATE TRIGGER`
  - used to specify actions the database system performs when certain events and conditions occur.

SQL

- Each assertion is given a constraint name
- For example, to specify the constraint that *the salary of an employee must not be greater than the salary of the manager of the department that the employee works for*:



```
CREATE ASSERTION SALARY_CONSTRAINT
CHECK ( NOT EXISTS   ( SELECT      *
                       FROM        EMPLOYEE E, EMPLOYEE M,
                                   DEPARTMENT D
                       WHERE       E.Salary>M.Salary
                                   AND E.Dno=D.Dnumber
                                   AND D.Mgr_ssn=M.Ssn ) );
```

- Whenever a tuple causes the condition to evaluate to FALSE, the constraint is violated.

SQL

Trigger

- A `trigger` is a statement the system executes automatically when event occurs as a side effect of a modification to the database.
- To design a trigger mechanism:
  1. Specify when a trigger is to be executed.
  2. Specify actions to be taken.
- execution is responsibility of the database system.

SQL

CREATE TRIGGER

- Check whenever an employee's salary is greater than the salary of direct supervisor.
- Triggered by:

SQL

CREATE TRIGGER

- Check whenever an employee's salary is greater than the salary of direct supervisor.
- Triggered by:
  - inserting a new employee
  - changing an employee's salary
  - changing an employee's supervisor.

SQL

CREATE TRIGGER

```
CREATE TRIGGER SALARY_VIOLATION
BEFORE INSERT OR UPDATE OF SALARY, SUPERVISOR_SSN
ON EMPLOYEE
FOR EACH ROW
    WHEN ( NEW.SALARY > ( SELECT SALARY FROM EMPLOYEE
                          WHERE SSN = NEW.SUPERVISOR_SSN ) )
                          INFORM_SUPERVISOR(NEW.Supervisor_ssn, NEW.Ssn );
```

SQL

CREATE TRIGGER

- A typical trigger has three components:
  1. event: database update operations.
     - make sure all events are accounted for.
     - specified after BEFORE or AFTER.
  2. condition that determines whether the rule action should be executed
     - specified in the WHEN clause of the trigger.
     - if no condition is specified, the action will be executed.
  3. action to be taken.

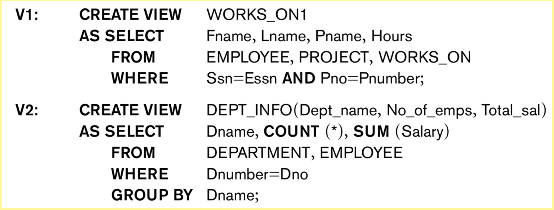
Views

Views

- A `view` is a single table that is derived from other tables.
- a way of specifying a table that we need to reference frequently, even though it may not exist physically.
- to specify a view use `CREATE VIEW`
  - a name
  - a list of attribute names
  - a query to specify the contents of the view.

Views

Views

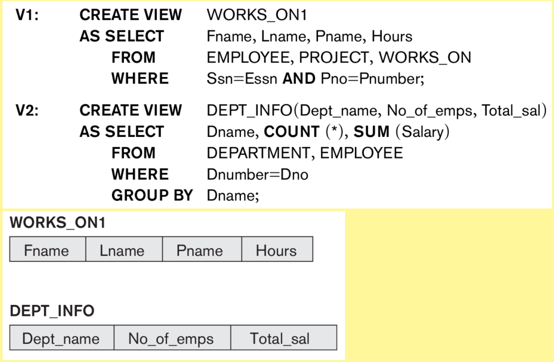


```
V1:     CREATE VIEW   WORKS_ON1
        AS SELECT     Fname, Lname, Pname, Hours
           FROM       EMPLOYEE, PROJECT, WORKS_ON
           WHERE      Ssn=Essn AND Pno=Pnumber;

V2:     CREATE VIEW   DEPT_INFO(Dept_name, No_of_emps, Total_sal)
        AS SELECT     Dname, COUNT (*), SUM (Salary)
           FROM       DEPARTMENT, EMPLOYEE
           WHERE      Dnumber=Dno
           GROUP BY   Dname;
```

Views

Views



```
V1:   CREATE VIEW   WORKS_ON1
      AS SELECT     Fname, Lname, Pname, Hours
         FROM       EMPLOYEE, PROJECT, WORKS_ON
         WHERE      Ssn=Essn AND Pno=Pnumber;

V2:   CREATE VIEW   DEPT_INFO(Dept_name, No_of_emps, Total_sal)
      AS SELECT     Dname, COUNT (*), SUM (Salary)
         FROM       DEPARTMENT, EMPLOYEE
         WHERE      Dnumber=Dno
         GROUP BY   Dname;
```

WORKS_ON1

Fname	Lname	Pname	Hours

DEPT_INFO

Dept_name	No_of_emps	Total_sal

Views

Views

- A view is always *up-to-date*
  - if base tables are modified the view must reflect the changes.
  - view is materialized when the query is executed.
  - responsibility of the DBMS
- we can use the `DROP VIEW` command to remove a view

```
DROP VIEW WORKS_ON1;
```

Views

Views

The problem of efficiently implementing a view for querying is complex

- query modification, transforms the view query into a query on the real tables.

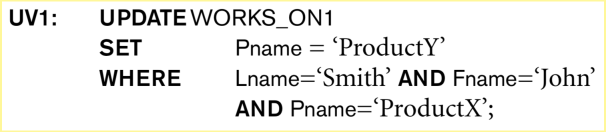
  ```
  SELECT Fname, Lname
  FROM EMPLOYEE, PROJECT, WORKS_ON
  WHERE Ssn=Essn AND Pno=Pnumber
                 AND Pname='ProductX';
  ```

- view materialization, involves physically creating a temporary view table when the view is first queried and keeping that table on the assumption that other queries on the view will follow.

Views

View Updates

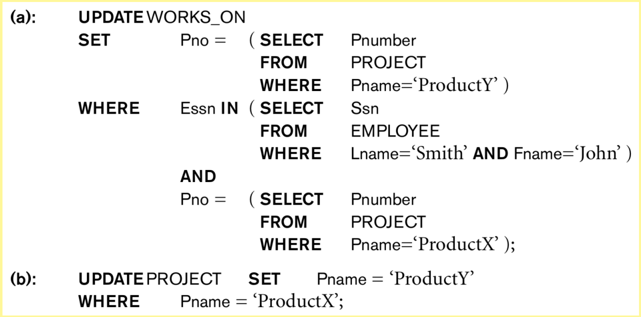
- Updating of views is complicated and can be ambiguous.
- An update on a view of a single table can be mapped to an update on the underlying base table.
- If a view involves joins, an update operation may be mapped in multiple ways.



```
UV1:    UPDATE  WORKS_ON1
        SET     Pname = 'ProductY'
        WHERE   Lname='Smith' AND Fname='John'
                AND Pname='ProductX';
```

Views

View Updates



```
(a):    UPDATE WORKS_ON
        SET       Pno =    ( SELECT   Pnumber
                             FROM     PROJECT
                             WHERE    Pname='ProductY' )
        WHERE     Essn IN  ( SELECT   Ssn
                             FROM     EMPLOYEE
                             WHERE    Lname='Smith' AND Fname='John' )
                  AND
                  Pno =    ( SELECT   Pnumber
                             FROM     PROJECT
                             WHERE    Pname='ProductX' );

(b):    UPDATE PROJECT    SET      Pname = 'ProductY'
        WHERE     Pname = 'ProductX';
```

Views

View Updates

- Only one possible update on the base relations can accomplish the desired update effect on the view.
- In general:
    - A view on a single table is updatable if the view contains the PK of the base relation, and attributes with the `NOT NULL` constraint with no default values.
    - Views on multiple tables using joins are generally not updatable.
    - Views using grouping and aggregate functions are not updatable.
- In SQL, the clause `WITH CHECK OPTION` must be added at the end.

Questions?